CEN

WORKSHOP

AGREEMENT

CWA 16008-4

August 2009

ICS 35.240.40

English version

J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Part 4: Text Input/Output Device Class Interface - Programmer's Reference

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

© 2009 CEN All rights of exploitation in any form and by any means reserved worldwide for CEN national Members.

Ref. No.:CWA 16008-4:2009 E

Contents

Foreword

This CWA contains the specifications that define the J/eXtensions for Financial Services (J/XFS) for the Java TM Platform, as developed by the J/XFS Forum and endorsed by the CEN J/XFS Workshop. J/XFS provides an API for Java applications which need to access financial devices. It is hardware independent and, by using 100% pure Java, also operating system independent.

The CEN J/XFS Workshop gathers suppliers (among others the J/XFS Forum members), service providers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN Secretariat, and at http://www.cen.eu/cenorm/sectors/sectors/isss/activity/jxfs_membership.asp. The specification was agreed upon by the J/XFS Workshop Meeting of 2009-05-6/9 in Brussels, and the final version was sent to CEN for publication on 2009-06-12.

The specification is continuously reviewed and commented in the CEN J/XFS Workshop. The information published in this CWA is furnished for informational purposes only. CEN makes no warranty expressed or implied, with respect to this document. Updates of the specification will be available from the CEN J/XFS Workshop public web pages pending their integration in a new version of the CWA (see http://www.cen.eu/cenorm/sectors/sectors/isss/activity/jxfs_cwas.asp).

The J/XFS specifications are now further developed in the CEN J/XFS Workshop. CEN Workshops are open to all interested parties offering to contribute. Parties interested in participating and parties wanting to submit questions and comments for the J/XFS specifications, please contact the J/XFS Workshop Secretariat hosted in CEN (jxfs-helpdesk@cen.eu).

Questions and comments can also be submitted to the members of the J/XFS Forum through the J/XFS Forum web-site http://www.jxfs.net.

This CWA is composed of the following parts:

- Part 1: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Base Architecture - Programmer's Reference
- Part 2: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Pin Keypad Device Class Interface - Programmer's Reference
- Part 3: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Magnetic Stripe & Chip Card Device Class Interface - Programmer's Reference
- Part 4: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Text Input/Output Device Class Interface - Programmer's Reference
- Part 5: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Cash Dispenser, Recycler and ATM Device Class Interface - Programmer's Reference
- Part 6: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Printer Device Class Interface - Programmer's Reference
- Part 7: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Alarm Device Class Interface - Programmer's Reference
- Part 8: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Sensors and Indicators Unit Device Class Interface - Programmer's Reference
- Part 9: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Depository Device Class Interface - Programmer's Reference
- Part 10: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Check Reader/Scanner Device Class Interface - Programmer's Reference (deprecated in favour of Part 13)
- Part 11: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Camera Device Class Interface - Programmer's Reference
- Part 12: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Vendor Dependant Mode Specification - Programmer's Reference
- Part 13: J/eXtensions for Financial Services (J/XFS) for the Java Platform – Scanner Device Class Interface - Programmer’s Reference (recommended replacement for Part 10)

Note: Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. The Java Trademark Guidelines are currently available on the web at http://www.sun.com All other trademarks are trademarks of their respective owners.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI.

Comments or suggestions from the users of the CEN Workshop Agreement are welcome and should be addressed to the CEN Management Centre.

History

Main differences to CWA 14923-4:2004 are:

- Two new keyboard data capabilities: JXFS_TIO_KEY_CONTROL and JXFS_TIO_KEY_MISC introduced.
- Keys which are supported by the readKeyboardData job modified.
- New getSupportedKeys job to retrieve what keys are supported from a keyboard capability subset.
- Description of existing readKeyboardData has been enhanced to clarify which keys are termination keys and handling of 00 (double zero) and 000 (triple zero) keys.
- new readKeyboardData job to enable default field value and new capability property readKeyboardDataWithDefault to know if it is supported.
- new getKeys job to retrieve available keys from the keyboard. Also new getKeysSupported capability added to check if new job is supported.
- open job handling clarified at base architecture level so specific chapter in this document is removed.
- status property has been deprecated as a roadmap for future enhancement, according to the new status design defined by base architecture documentation.
- specific declaration of result codes used by each job has been removed, and now result refers to common section at the end of the document.
- New JxfsTIOStatusSelectorEnum enumeration introduced to allow use of new getStatus method defined in base architecture documentation.
- Constant Definitions have been added at the end of the document.

Main differences to CWA 13937-4:2000 are:

- Modified the description for the readKeyboardData method.
- Changed the description for the readKeyboardData flush parameter.
- Changed the description for the readKeyboardData autoEnd parameter.
- Added paragraph specifying handling of null parameters.
- Added a class hierarchy diagram
- Removed the JXFS_E_CLAIMED exception
- Modified the Description of the IJxfsTIOControl's resolutionProperty
- Modified the Description of the IJxfsTIOControl's beep method
- Modified the Description of the IJxfsTIOControl's getLED method
- Modified the Description of the IJxfsTIOControl's clearScreen method
- Modified the Description of the IJxfsTIOControl's writeDisplayData method
- Modified the Description of the IJxfsTIOControl's readKeyboardData method
- Modified the comment for the parameter "numOfChars" passed into the readKeyboardData call
- Modified the comment for the field "data" of the OperationCompleteEvent of the readKeyboardData
- Modified the access of the JxfsTIOStatus's properties

1. Scope

This document describes the Text Input / Output Device Class (TIO) based on the basic architecture of J/XFS which is similar to the JavaPOS architecture. It is event driven and asynchronous.

Three basic levels are defined in JavaPOS. For J/XFS this model is extended by a communication layer, which provides device communication that allows distribution of applications and devices within a network. So we have the following layers in J/XFS:

- Application
- Device Control and Manager
- Device Communication
- Device Service

Application developers program against control objects and the Device Manager which reside in the Device Control Layer. This is the usual interface between applications and J/XFS Devices. Device Control Objects access the Device Manager to find an associated Device Service. Device Service Objects provide the functionality to access the real device (i.e. like a device driver).
During application startup the Device Manager is responsible for locating the desired Device Service Object and attaching this to the requesting Device Control Object. Location and/or routing information for the Device Manager reside in a central repository.

To support Text I/O Devices, the basic Device Control structure is extended with various properties and methods specific to this device which are described on the following pages.

2. Overview

2.1. Description

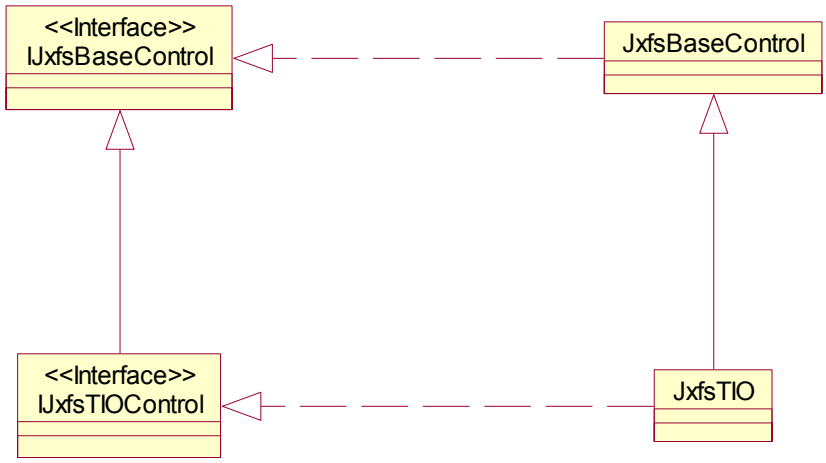
The Text Input Output Device Control class, defined in the *JxfsTIO* class, is a subclass of the *JxfsBaseControl* class. Its interface is defined in the *IJxfsTIOControl* interface which is a subclass of the *IJxfsBaseControl* class. The intended use of an Text Input Output object is to allow data and control to be passed between a Java application or applet and a TIO type device so that the associated device can be accessed through a "Pure Java" platform.

As stated previously, the Text Input Output Device Control class allows access to TIO type devices. An overview of the device operation is described in this section from the point of view of the application or applet (referred to as just an application).

An application will instantiate a *JxfsTIO* object and then use the available methods to do I/O. If an error occurs in initiating the I/O, an *JxfsException* will be thrown. The application should be designed to catch and handled the errors thrown. Otherwise, control will be returned to the application and a *JxfsEvent* will be used to signal I/O completion asynchronous to the invoking applications thread of execution.
As a result of the event based I/O operation model, an application will have to register itself as a listener with the *JxfsTIO* object for the event(s) generated.

This document describes the input and output features of the TIO. It offers the functionality of a text display, a set of LEDs and a beep mechanism. In addition function keys and a tiny keyboard are also supported.

2.2. Class Hierarchy



2.3. Classes and Interfaces

Class or interface	Name	Description	Extends / Implements
Interface	**IJxfsBaseControl**	Base interface for all device controls. Contains methods specific to all the device controls.	—
Class	**JxfsBaseControl**	Base class for all device controls. Implements methods common for all devices.	—
Interface	**IJxfsTIOControl**	Base interface for all Text Input/Output controls. Contains the methods specific to all the device controls for the Text Input/Output device category.	Extends: **IJxfsBaseControl**
Interface	**IJxfsTIOService**	Base interface for all Text Input/Output services. Contains the methods specific to all the device services for the Text Input/Output device category.	Extends: **IJxfsBaseService**
Class	**JxfsTIO**	Base class for all Text Input/Output controls Implements the methods defined in the ***IJxfsTIOControl*** Interface. Contains the properties specific to all Text Input/Output device controls.	Extends: **JxfsBaseControl** Implements: **IJxfsTIOControl**

2.4. Support Classes

Class or interface	Name	Description	Extends / Implements
Interface	**JxfsConst**	Interface containing the J/XFS constants that are common to several device categories	—
Interface	**IJxfsTIOConst**	Interface containing the J/XFS constants that are common to all the Text Input/Output device controls.	—
Class	**JxfsTIOStatus**	Describes the TIO specific status information.	Extends: **JxfsStatus**
Class	**JxfsTIOResolution**	Keeps the resolution (in characters per row and column).	Extends: **JxfsType**

3. Device behavior

3.1. Handling of null parameters

If *null* is passed as a method parameter, a *JxfsException* exception with the *errorCode* property set to JXFS_E_PARAMETER_INVALID will be thrown, unless the handling of a *null* parameter is explicitly specified for a particular method.

4. Classes and Interfaces

4.1. IJxfsTIOControl

4.1.1. Introduction

The J/XFS Text Input/Output Device Control interface is defined in *IJxfsTIOControl* and extends of *IJxfsBaseControl*. The intent of the J/XFS Text Input/Output Device Control object is to allow data and control to pass between the application and the device support code so that the associated device can be accessed.

4.1.2. Summary

Please note the following when determining the meaning of a property's
Access:
R The property is read only.
W The property is write only.
R/W The property may be read or written.

To read or write a property send the J/XFS Text Input/Output Device Control object the appropriate JavaBeans conform method.

It should not be assumed that the device has a clear screen after an open.

Extends: IJxfsTIOControl

Properties

Property	Type	Access	Initialized by
cursorSupported	boolean	R	device service
status – deprecated	JxfsTIOStatus	R	device service
resolution	JxfsTIOResolution	R/W	device service
availableResolutions	Vector	R	device service
displayLightSupported	boolean	R	device service
beepSupported	boolean	R	device service
maxLED	int	R	device service
keyboardSupported	boolean	R	device service
keyboardLockSupported	boolean	R	device service
getKeysSupported	boolean	R	device service
readKeyboardDataWithDefault	boolean	R	device service

Methods

Method	Return	Meaning
beep	int	Sounds a beep signal.
lightDisplay	int	Lights the text display.
setLED	int	Lights the specified LED.
getLED	int	Gets the current light type of specified LED.
clearScreen	int	Clears display screen.
writeDisplayData	int	Writes data on display.
isTextAttributeSupported	boolean	Detects supported text attributes.
readKeyboardData	int	Reads pressed keys.
getKeys	int	Wait for the pressed keys in raw mode.
getSupportedKeys	int	Defines what keys are supported from a keyboard capability subset

The common exceptions thrown by all methods are:

Value	Meaning
JXFS_E_CLOSED	The Device Control has not been opened.
JXFS_E_UNREGISTERED	The device is not registered at the JxfsDeviceManager
JXFS_E_REMOTE	A network error occurred
JXFS_E_PARAMETER_INVALID	Parameter passed to method is invalid.
JXFS_E_NOT_SUPPORTED	Method is not supported.

4.1.3. Properties

cursorSupported Property R

Type *boolean*
Initial Value -
Description Specifies whether the Text Input Output device has a display cursor. The value can be *true* or *false* depending on the characteristics of the display.
Events No additional events generated.
Exceptions No additional exceptions thrown.

status Property R

Type *JxfsTIOStatus*
Initial Value -
Description **Deprecated** - Depending on the state of the Text Input Output device, the status object will be updated. For details on *JxfsTIOStatus* please see the appropriate section.
Events If the values of these properties kept by the status object changes the device service will send all registered *JxfsStatusEvent* listeners a *JxfsStatusEvent* with *status* = JXFS_S_TIO_STATUS_CHANGED. The status object is attached in the *details* field.
Exceptions No additional exceptions thrown.

resolution Property R/W

Type *JxfsTIOResolution*
Initial Value -
Description Specifies the horizontal and vertical size of the display in character columns and rows. If no resolution is set or an unsupported resolution is specified the default resolution will be used. After redefining resolution and before displaying a new text the display should be cleared to assure proper text output. If no screen is available a resolution of 0,0 is returned.
Events No additional events generated.
Exceptions No additional exceptions thrown.

availableResolutions R

Type *Vector*
Initial Value -
Description Specifies available display resolutions. All resolutions are kept in a *Vector* consisting of *JxfsTIOResolution* objects.
Events No additional events generated.
Exceptions No additional exceptions thrown.

displayLightSupported Property R

Type *boolean*
Initial Value -
Description Specifies whether the Text Input Output device supports display light. The value can be true or false depending on the characteristics of the device.
Events No additional events generated.
Exceptions No additional exceptions thrown.

beepSupported Property R

Type *boolean*
Initial Value -
Description Specifies whether the Text Input Output device supports beeping. The value can be true or false depending on the characteristics of the device.
Events No additional events generated.
Exceptions No additional exceptions thrown.

maxLED Property R

Type *int*
Initial Value -
Description Specifies the number of LED‘s supported by the Text Input Output device.
Events No additional events generated.
Exceptions No additional exceptions thrown.

keyboardSupported Property R

Type *boolean*
Initial Value -
Description Specifies if a keyboard is supported. The value is *true* if available, *false* otherwise.
Events No additional events generated.
Exceptions No additional exceptions thrown.

keyboardLockSupported Property R

Type *boolean*
Initial Value -
Description Specifies if a keyboardLock is supported. The value is *true* if available, *false* otherwise.
Events No additional events generated.
Exceptions No additional exceptions thrown.

getKeysSupported Property R

Type *boolean*
Initial Value -
Description Specifies if the *getKeys()* method is supported by the device service.
Events No additional events generated.
Exceptions No additional exceptions thrown.

readKeyboardDataWithDefault Property R

Type *boolean*
Initial Value -
Description Defines whether this Device Service supports the *readKeyboardData()* with the defaultInput parameter.
Events No additional events generated.
Exceptions No additional exceptions thrown.

4.1.4. Methods

beep

Syntax ***int beep(int beepValue, int time) throws JxfsException;***

Description This method can be used to set the conditions for sounding a beep (in case of _BEEP_KEYPRESS) or for actually sounding a beep. Returns an *identificationID* that identifies this operation.

beepValue can be one of the following:

Value	Meaning
JXFS_TIO_BEEP_OFF	The beeper is turned off.
JXFS_TIO_BEEP_KEYPRESS	The beeper will sound on key press.
JXFS_TIO_BEEP_CONTINUOUS	The beeper sounds continuously.
JXFS_TIO_BEEP_EXCLAMATION	The beeper sounds an exclamation signal.
JXFS_TIO_BEEP_WARNING	The beeper sounds an warning signal.
JXFS_TIO_BEEP_ERROR	The beeper sounds an error signal.
JXFS_TIO_BEEP_CRITICAL	The beeper sounds an critical error signal.

time in milliseconds. If the value is greater than zero the TIO will beep for the specified time. If equal to JXFS_FOREVER, beeping is performed forever. If *beep()* is called a second time the current beeping ends always immediately (e.g. with *beepValue* equal to JXFS_TIO_BEEP_OFF or with a new specified time). If this method is called with a *beepValue* of JXFS_TIO_BEEP_OFF the time parameter is ignored and key press beeping will also be switched off (if it was on).
A time of JXFS_FOREVER is valid for the *beepValue* of JXFS_TIO_BEEP_KEYPRESS.

The operation complete event for this method will return immediately once the beep has been initiated at the device level. The reasons for this approach are:

If you specify an infinite time, an operation complete event will never return if the beep is not cancelled or another beep is not issued.

If you assume that operation compete events are issued after the completion of the beep, then this might also have an impact on the scheduling of commands. If you have a simple scheduling implementation where the next scheduled command will be run after the previous has finished, then you could never stop a beep with an infinite time with another beep.

There are many peripheral devices that themselves take beep commands with a time constraint. For these devices implementations are a lot easier, if you issue the beep command to the device and let the device do the rest; but only very few terminals have the ability to return information whether they are still beeping or not. So it is easier, in many cases, for implementations if the operation complete event is issued earlier than the end of the beep.

Therefore, it is assumed that the operation complete event for this method indicates whether the beep was initiated successfully; and does not reflect whether or not it was cancelled by a succeeding beep method call.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an *JxfsOperationCompleteEvent* having a status value of:

Field	Value & Meaning
operationID	JXFS_O_TIO_BEEP
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

lightDisplay

Syntax ***int lightDisplay(boolean on) throws JxfsException;***

Description This method can be used to switch display lighting on (***on*** equals true) or off (***on*** equals false). Returns an *identificationID* that identifies this operation.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an *JxfsOperationCompleteEvent* having a status value of:

Field	Value & Meaning
operationID	JXFS_O_TIO_LIGHT
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

setLED

Syntax ***int setLED(int index, int type) throws JxfsException;***

Description This method can be used for lighting a LED. It returns an *identificationID* that identifies this operation.

type can be one of the following:

Value	Meaning
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.

index Specifies which LED to light. If it is equal to a value from 1 to *maxLED* the LED with the appropriate index will be lighted. In addition to specifying the number of the LED it can be equal to one of the following values:

Value	Meaning
JXFS_TIO_LED_ERROR	The error LED will be lighted.
JXFS_TIO_LED_WARNING	The warning LED will be lighted.
JXFS_TIO_LED_ONLINE	The online LED will be lighted.
JXFS_TIO_LED_OFFLINE	The offline LED will be lighted (or the online LED turns off).
JXFS_TIO_LED_NORMAL	Indicates proper working of the device
JXFS_TIO_LED_PAPERLOW	The paper low LED will be lighted.
JXFS_TIO_LED_PAPEREMPTY	The paper empty LED will be lighted.
JXFS_TIO_LED_PAPERJAM	The paper jam LED will be lighted.

JXFS_TIO_LED_TONERLOW	The toner low LED will be lighted.
JXFS_TIO_LED_TONEREMPTY	The toner empty LED will be lighted.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an *JxfsOperationCompleteEvent* having a status value of:

Field	**Value & Meaning**
operationID	JXFS_O_TIO_LED
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

getLED

Syntax ***int getLED(int index) throws JxfsException;***

Description This method can be used to query the current lighting of an LED. Returns a type code specifying the lightning status. Throws an exception JXFS_E_PARAMETER_INVALID. The valid values of the *index* parameter are the same as those specified in the settled method.

The returned integer is one of the following:

Value	**Meaning**
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.

clearScreen

Syntax ***int clearScreen(int positionX, int positionY, int width, int height) throws JxfsException;***

Description This method can be used to clear the display screen. All parameters are in column positions. Returns an *identificationID* that identifies this operation.

The *positionX* and *positionY* parameters indicate the top left corner of the area to be cleared. A one based co-ordinate system where the display origin is at (1,1) in the top left of the display is used, with X and Y co-ordinates increasing to the right and down respectively. The current position of the cursor after the clear screen operation is the same as it was before this operation.

positionX	specifies the starting horizontal position of the area to be cleared.
positionY	specifies the starting vertical position of the area to be cleared.
width	specifies the horizontal width of the area to be cleared.
height	specifies the vertical height of the area to be cleared.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an *JxfsOperationCompleteEvent* having a status value of:

Field	**Value & Meaning**
operationID	JXFS_O_TIO_CLEAR
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

writeDisplayData

Syntax ***int writeDisplayData(int mode, int posX, int posY, int textAttr, java.lang.String text) throws JxfsException;***

Description This method can be used to write text to the display. The text is wrapped automatically on the end of the line, except on the last one, where text is truncated. Returns an *identificationID* that identifies this operation.

When relative positioning is selected *posY* and *posX* can be 0, meaning write at the current output position. If *posY* or *posX* are less than 0 output is written above or to the left of the current output position respectively.

When writing more data then can be displayed on the last display line, the data being output is truncated and the output position for the next write command is set to the origin (top left) of the display.

mode Specifies the mode of text positioning. It can be one of the following:

Value	Meaning
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

posX Specifies the starting horizontal position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode, where value 1 means the most left position.

posY Specifies the starting vertical position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode, where value 1 means the most top position.

textAttr Specifies the text attributes of the text being displayed. It can have a combination of the following values:

Value	Meaning
JXFS_TIO_TEXT_NORMAL	The normal text display.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

If one of the modi mentioned above is not supported, another best matching mode is selected. The values can also be combined (i.e. underline and inverted). This is achieved by OR'ing the corresponding values.

text Specifies the text to be displayed.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an *JxfsOperationCompleteEvent* having a status value of:

Field	Value & Meaning
operationID	JXFS_O_TIO_DISPLAY
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

readKeyboardData

Syntax ***int readKeyboardData(int numOfChars, int mode, int posX, int posY, int echoMode, int echoAttr, int keys, boolean cursor, boolean flush, boolean autoEnd) throws JxfsException***

Description This method can be used to read unformatted text from the keyboard. When complete a *Vector* containing the keys pressed will be placed in the *data* field of a *JxfsOperationCompleteEvent* and all *JxfsOperationCompleteEvent* listeners will be notified. Returns an *identificationID* that identifies this operation.

If the user enters only the Return key, and text is displayed in the entry field by a preceding *writeDisplayData()* call, the returned data in the *JxfsOperationCompleteEvent* will contain no data. If the user enters a key, other than the Return key, and *echoMode* is set to JXFS_TIO_ECHO_TEXT or JXFS_TIO_ECHO_PASSWORD, the display will be cleared for the *numOfChars* specified, and only the keys entered will be displayed.

All function keys, the ENTER key, the CANCEL key and the HELP key are terminate keys. The CLEAR key will cause all input to be cleared on the TTU and in the input buffer. The CLEAR key will not appear in the input buffer. If input is terminated by a function or cancel key, the terminating key is appended to the output data to allow analysis of the termination reason. Therefore, the description of valid keys contained in the data member of the operation complete events returned for this method includes the JXFS_TIO_KEY_CANCEL value.

The 00 (double zero) and the 000 (triple zero) keys are replaced in the input buffer by two or three 0 (zero) characters. If the input field (ie size defined by *numOfChars*) is too small to accommodate 2 or three such characters, the input is truncated. The DELETE key will only delete one character from the input buffer at a time. The double zero and triple zero keys result in two or three keys being echoed on the TTU if used in TEXT or PASSWORD modes (though use of these characters in PASSWORD mode should be discouraged).

The *posX* and *posY* parameters indicate the top left corner of the output position. When absolute positioning is used a one based co-ordinate system, where the display origin is at (1,1) in the top left of the display is used, with X and Y co-ordinates increasing to the right and down respectively. When relative positioning is selected *posY* and *posX* can be 0, meaning write at the current output position. If *posY* or *posX* are less than 0 output is written above or to the left of the current output position respectively. If *echoMode* is JXFS_TIO_ECHO_INVISIBLE then the output position is not adjusted as no text is echoed to the display.

When writing to the last display line data being output is truncated and the output position is set to the origin (top left) of the display.

numOfChars Specifies the number of characters to be read from the keyboard. This parameter does not include any depressions of the delete or backspace keys.

mode Specifies the mode of text positioning. It can be one of the following:

Value	Meaning
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

posX Specifies the starting horizontal position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

posY Specifies the starting vertical position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

echoMode Specifies the text attributes of the input being echoed. It can have one of the following values:

Value	Meaning
JXFS_TIO_ECHO_TEXT	The input will be echoed.
JXFS_TIO_ECHO_INVISIBLE	The input will not be echoed.
JXFS_TIO_ECHO_PASSWORD	The input will echo a

replacement character.

echoAttr Specifies the text attributes of the text being echoed. It can have a combination of the following values:

Value	Meaning
JXFS_TIO_TEXT_NORMAL	The normal text display.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

If one of the modi mentioned above is not supported, another best matching mode is selected. The values can also be combined (i.e. underline and inverted). This is achieved by OR'ing the corresponding values.

keys Specifies what types of keys the keyboard of the Text Input Output device will allow for input. It may have a value of a combination of the following:

Value	Meaning
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.
JXFS_TIO_KEY_CONTROL	The TIO has control keys.
JXFS_TIO_KEY_MISC	The TIO has miscellaneous keys.

The values can also be combined. This is achieved by OR'ing the corresponding values.

cursor Specifies whether the Text Input Output device will display a cursor. The value can be true or false depending on the characteristics of the display.

flush Specifies whether the Text Input Output device input buffer will be cleared before input is allowed.

autoEnd Specifies whether input is automatically ended by Device Services when the value given in *numOfChars* is met. If this is false the input is only ended by pressing the Enter key, the Cancel key or any function key. Note only the first x number of characters will be displayed on the TIO device and added to the input data, where *x=numOfChars*. Subsequent characters entered will not be displayed, and will be ignored, unless they are the delete key, in which case individual characters will be deleted from the input data. The return code is always successful, even if the number of keys entered is not equal to the *numOfChars* specified to be read..

Events

JxfsOperationCompleteEvent When a *readKeyboardData()* operation is completed a *JxfsOperationCompleteEvent* will be sent by the J/XFS TIO Device Control to all registered *JxfsOperationCompleteEvent* listeners. The *JxfsOperationCompleteEvent* will contain the following:

Field	Value & Meaning
operationID	JXFS_O_TIO_READ
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	A *Vector* of Integer objects containing the keys read. This does not contain the final Enter or any Delete keys pressed in between.

The following keys are supported:

Value	Meaning
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT	The (.) sign.
JXFS_TIO_KEY_COMMA	The (,) sign.
JXFS_TIO_KEY_SEMICOLON	The (;) sign.
JXFS_TIO_KEY_FENCE	The (#) sign.

JXFS_TIO_KEY_MULTI	The (*) sign.
JXFS_TIO_KEY_SLASH	The (/) sign.
JXFS_TIO_KEY_PLUS	The (+) sign.
JXFS_TIO_KEY_MINUS	The (-) sign.
JXFS_TIO_KEY_F1 ... F10	The function keys.
JXFS_TIO_KEY_DELETE and JXFS_TIO_KEY_CLEAR	Two of the control keys.

Exceptions No additional exceptions thrown.

readKeyboardData

Syntax ***int readKeyboardData(java.lang.String defaultInput, int numOfChars, int mode, int posX, int posY, int echoMode, int echoAttr, int keys, boolean cursor, boolean flush, boolean autoEnd) throws JxfsException***

Description This method can be used to read unformatted text from the keyboard. When complete a *Vector* containing the keys pressed will be placed in the *data* field of a *JxfsOperationCompleteEvent* and all *JxfsOperationCompleteEvent* listeners will be notified. Returns an *identificationID* that identifies this operation.

If the user enters only the Return key, and a default input is defined, the returned data in the *JxfsOperationCompleteEvent* will contain the default input. If the user enters a key, other than the Return key, and echoMode is set to JXFS_TIO_ECHO_TEXT or JXFS_TIO_ECHO_PASSWORD, the display will be cleared for the *numOfChars* specified, and only the keys entered will be displayed.

All function keys, the ENTER key, the CANCEL key and the HELP key are terminate keys. The CLEAR key will cause all input to be cleared on the TTU and in the input buffer. The CLEAR key will not appear in the input buffer. If input is terminated by a function or cancel key, the terminating key is appended to the output data to allow analysis of the termination reason. Therefore, the description of valid keys contained in the data member of the operation complete events returned for this method includes the JXFS_TIO_KEY_CANCEL value.

The 00 (double zero) and the 000 (triple zero) keys are replaced in the input buffer by two or three 0 (zero) characters. If the input field (ie size defined by *numOfChars*) is too small to accommodate 2 or three such characters, the input is truncated. The DELETE key will only delete one character from the input buffer at a time. The double zero and triple zero keys result in two or three keys being echoed on the TTU if used in TEXT or PASSWORD modes (though use of these characters in PASSWORD mode should be discouraged).

The *posX* and *posY* parameters indicate the top left corner of the output position. When absolute positioning is used a one based co-ordinate system, where the display origin is at (1,1) in the top left of the display is used, with X and Y co-ordinates increasing to the right and down respectively. When relative positioning is selected *posY* and *posX* can be 0, meaning write at the current output position. If *posY* or *posX* are less than 0 output is written above or to the left of the current output position respectively. If *echoMode* is JXFS_TIO_ECHO_INVISIBLE then the output position is not adjusted as no text is echoed to the display.

When writing to the last display line data being output is truncated and the output position is set to the origin (top left) of the display.

defaultInput Specifies a default string to be displayed in the input field and initializes the input buffer with it. If the user enters no input (ENTER only), this input buffer is returned in the *JxfsOperationCompleteEvent* data field. If *defaultInput* is empty or a null reference, the input buffer is not initialized and is a valid

parameter. If the defaultInput is longer than *numOfChars* or contains characters not allowed by the keys parameter, the JXFS_E_INVALID_PARAMETER exception is thrown.

numOfChars Specifies the number of characters to be read from the keyboard. This parameter does not include any depressions of the delete or backspace keys.

mode Specifies the mode of text positioning. It can be one of the following:

Value	Meaning
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

posX Specifies the starting horizontal position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

posY Specifies the starting vertical position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

echoMode Specifies the text attributes of the input being echoed. It can have one of the following values:

Value	Meaning
JXFS_TIO_ECHO_TEXT	The input will be echoed.
JXFS_TIO_ECHO_INVISIBLE	The input will not be echoed.
JXFS_TIO_ECHO_PASSWORD	The input will echo a replacement character.

echoAttr Specifies the text attributes of the text being echoed. It can have a combination of the following values:

Value	Meaning
JXFS_TIO_TEXT_NORMAL	The normal text display.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

If one of the modi mentioned above is not supported, another best matching mode is selected. The values can also be combined (i.e. underline and inverted). This is achieved by OR'ing the corresponding values.

keys Specifies what types of keys the keyboard of the Text Input Output device will allow for input. It may have a value of a combination of the following:

Value	Meaning
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.
JXFS_TIO_KEY_CONTROL	The TIO has control keys.
JXFS_TIO_KEY_MISC	The TIO has miscellaneous keys.

The values can also be combined. This is achieved by OR'ing the corresponding values.

cursor Specifies whether the Text Input Output device will display a cursor. The value can be true or false depending on the characteristics of the display.

flush Specifies whether the Text Input Output device input buffer will be cleared before input is allowed. If flush is set to TRUE, the defaultvalue is not displayed and the input buffer not initialized with it.

autoEnd Specifies whether input is automatically ended by Device Services when the value given in *numOfChars* is met. If this is false the input is only ended by pressing the Enter key, the Cancel key, or any function key. Note only the first x number of characters will be displayed on the TIO device and added to the input data, where x=*numOfChars*. Subsequent characters entered will not be displayed, and will be ignored, unless they are the delete key, in which case

individual characters will be deleted from the input data. The return code is always successful, even if the number of keys entered is not equal to the *numOfChars* specified to be read.

Events

JxfsOperationCompleteEvent When a *readKeyboardData()* operation is completed a *JxfsOperationCompleteEvent* will be sent by the J/XFS TIO Device Control to all registered *JxfsOperationCompleteEvent* listeners. The *JxfsOperationCompleteEvent* will contain the following:

Field	Value & Meaning
operationID	JXFS_O_TIO_READ
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	A *Vector* of Integer objects containing the keys read.. This does not contain the final Enter or any Delete keys pressed in between.

The following keys are supported:

Value	Meaning
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT	The (.) sign.
JXFS_TIO_KEY_COMMA	The (,) sign.
JXFS_TIO_KEY_SEMICOLON	The (;) sign.
JXFS_TIO_KEY_FENCE	The (#) sign.
JXFS_TIO_KEY_MULTI	The (*) sign.
JXFS_TIO_KEY_SLASH	The (/) sign.
JXFS_TIO_KEY_PLUS	The (+) sign.
JXFS_TIO_KEY_MINUS	The (-) sign.
JXFS_TIO_KEY_F1 ... F10	The function keys.
JXFS_TIO_KEY_DELETE and JXFS_TIO_KEY_CLEAR	Two of the control keys.

Exceptions No additional exceptions thrown.

isTextAttributeSupported

Syntax ***boolean isTextAttributeSupported(int textAttr) throws JxfsException;***

Description This method is used to detect supported text attributes.

textAttr Specifies the text attribute the method is detecting. It can have a combination of the following values:

Value	Meaning
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

Events No additional events generated.

Exceptions No additional exceptions thrown.

getKeys

Syntax ***int getKeys(int bufferControl) throws JxfsException***

Description This method can be used to retrieve available (already pressed) keys from the keyboard. When completed the value of the read key(s) will be placed in the data field of a *JxfsOperationCompleteEvent* and all *JxfsOperationCompleteEvent* listeners will be notified. Returns an *identificationID* that identifies this operation.

Reading one or more keys with this method will not change or modify the display. This raw input mode will deliver all keys that have been pressed. The *bufferControl* parameter allows for the selection if keys shall also be reported that have been pressed before the operation has been started in the device service. See the following sequence diagrams for details.

bufferControl Specifies if the key buffer shall be cleared at the beginning of the method. It can be one of the following:

Value	Meaning
JXFS_TIO_CLEAR	The keyboard buffer will be cleared before waiting for a key. Even in this mode an application must be aware that this method may return more than one key.
JXFS_TIO_NOCLEAR	The keyboard buffer will not be cleared. If one or more keys are already present, they will be returned as result. Otherwise the method is waiting for a key to be pressed. The device service will return all pressed keys since the last *getKeys()*, *readKeyboardData()* or *open()* method call or since the last hardware error state. If a key has been entered before the *open()* method has been completed it is not guaranteed that it will be delivered.

Events

JxfsOperationCompleteEvent When a *getKeys()* operation is completed a *JxfsOperationCompleteEvent* will be sent by the J/XFS TIO Device Control to all registered *JxfsOperationCompleteEvent* listeners. The operation is completed

if at least one key is available, the operation has been cancelled or an error occurred. The *JxfsOperationCompleteEvent* will contain the following:

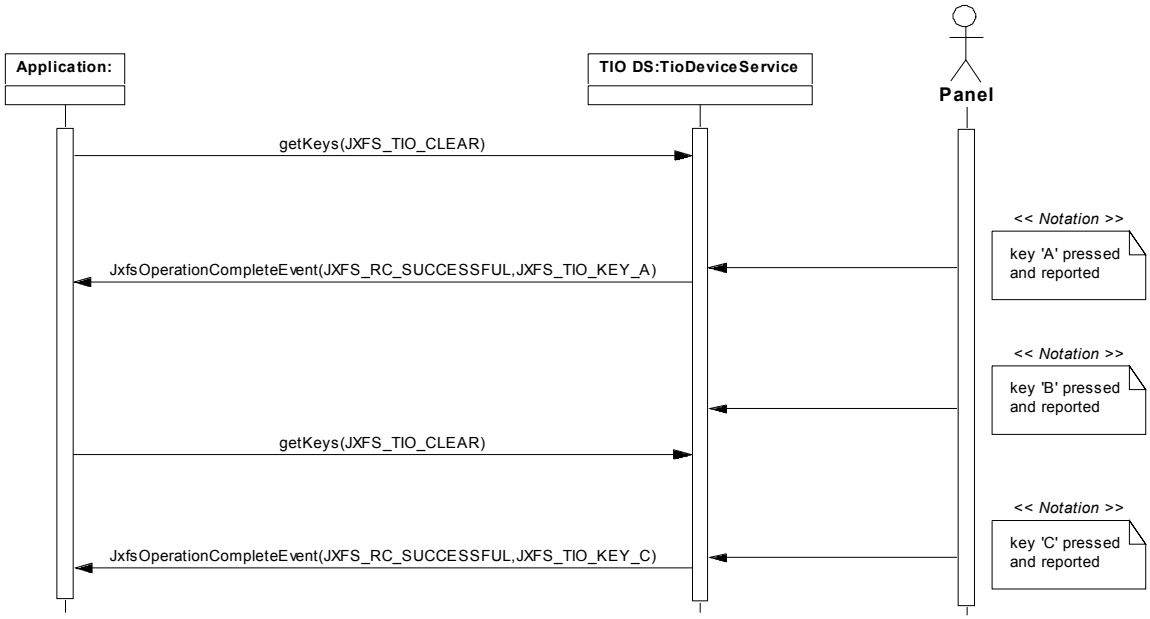
Field	**Value & Meaning**
operationID	JXFS_O_TIO_GETKEYS
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	Array of int values each specifying one read key. If no key has been read, data is null. An empty array is not allowed.

The following keys are supported:

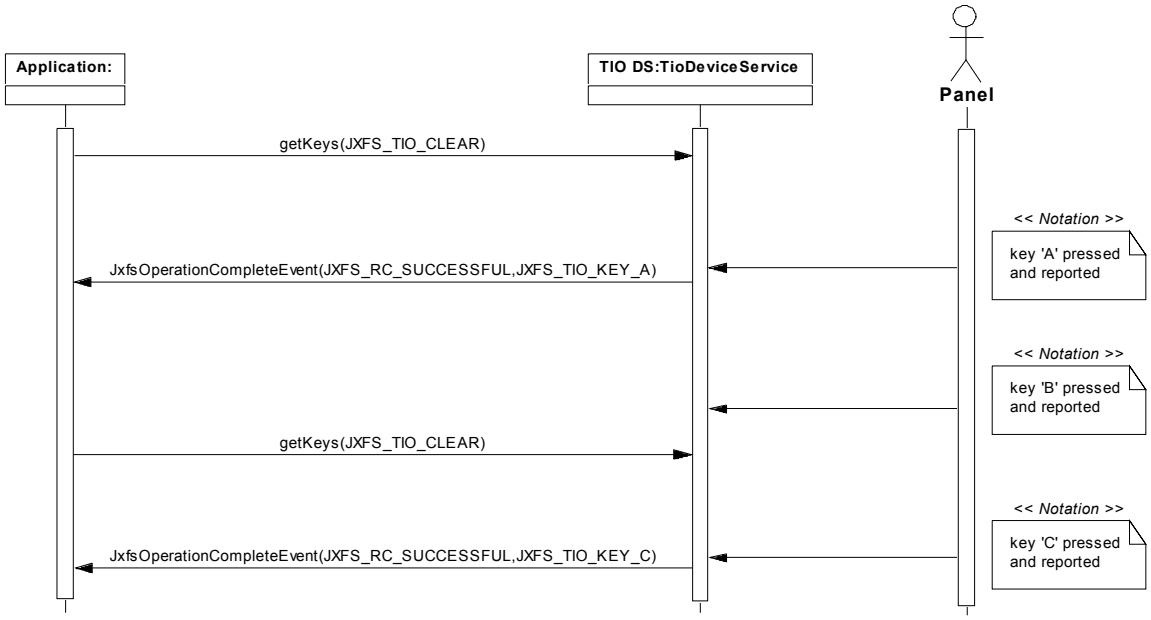
Value	**Meaning**
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT	The (.) sign.
JXFS_TIO_KEY_COMMA	The (,) sign.
JXFS_TIO_KEY_SEMICOLON	The (;) sign.
JXFS_TIO_KEY_FENCE	The (#) sign.
JXFS_TIO_KEY_MULTI	The (*) sign.
JXFS_TIO_KEY_SLASH	The (/) sign.
JXFS_TIO_KEY_PLUS	The (+) sign.
JXFS_TIO_KEY_MINUS	The (-) sign.
JXFS_TIO_KEY_DELETE	The delete key.
JXFS_TIO_KEY_CANCEL	The cancel key.
JXFS_TIO_KEY_ENTER	The enter key.
JXFS_TIO_KEY_F1 ... F10	The function keys.

Exceptions No additional exceptions thrown.

Enter a key with clearing the buffer:



Enter a key without buffer clearing:



getSupportedKeys

Syntax ***int getSupportedKeys(int capabilityType) throws JxfsException;***
Description Defines what keys are supported from a keyboard capability subset.

capabilityType Filter the keys returned according to this capability type. A value of zero means return all keys supported by the TIO device. The possible values may be a combination of the following flags:

Value	Meaning
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.
JXFS_TIO_KEY_CONTROL	The TIO has control keys.
JXFS_TIO_KEY_MISC	The TIO has miscellaneous keys.

Events

JxfsOperationCompleteEvent When a *getSupportedKeys()* operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with the following data:

Field	Value & Meaning
operationID	JXFS_O_TIO_GET_SUPPORTED_KEYS
identificationId	The corresponding Id for the completed operation.
result	Common or device dependent error code. (See Error codes on *Constants* section).
data	A *Vector* of *Integer(s)* which represent the keys supported within the capability type filter parameter. *null* if no keys are supported within the defined filter.

Exceptions No additional exceptions thrown.

4.2. IJxfsTIOService

The Device Service interface is common for all device services of this device type. It is used by the Device Controls to access the functionality of the device. This interface has to be implemented by any J/XFS Device Service.
The device type specific Device Service interface is similar to the Device Control interface. All device specific method calls are extended by an additional parameter (int control_id). This is always added as the last parameter in every operation.

4.3. JxfsTIO

This class is the implementation of the interface *IJxfsTIOControl*.

4.4. JxfsTIOStatus

4.4.1. Introduction

All TIO specific status informations are kept in the *JxfsTIOStatus* object, that can be queried by using the *getStatus()* method of the JxfsTIO class.

4.4.2. Summary

Extends: JxfsStatus

Property	Type	Access	Initialized by
JxfsTIOStatus()	constructor		-
set*Property(boolean value)* throws JxfsException	void		sets the corresponding property
is*Property()* throws JxfsException	boolean		gets the corresponding property
online	boolean	RW	device service
devicePresent	boolean	RW	device service
keyboardOn	boolean	RW	device service
keyboardLockOn	boolean	RW	device service

The constructor initializes all members to false.

4.4.3. Properties

online Property RW

Type *boolean*
Initial Value -
Description Returns *true* if the device is online, *false* if not.
Events No additional events generated.
Exceptions No additional exceptions thrown.

devicePresent Property RW

Type *boolean*
Initial Value -
Description Returns *true* if the device is attached to workstation and the power is on, *false* if not.
Events No additional events generated.
Exceptions No additional exceptions thrown.

keyboardOn Property RW

Type *boolean*
Initial Value -
Description Returns *true* if the keyboard is activated, *false* if not.
Events No additional events generated.
Exceptions No additional exceptions thrown.

keyboardLockOn Property RW

Type *boolean*
Initial Value -
Description Returns *true* if the keyboard lock is activated, *false* if not.
Events No additional events generated.
Exceptions No additional exceptions thrown.

4.5. JxfsTIOResolution

4.5.1. Introduction

This class keeps the resolution of the text display. The resolution is described as the number of characters that can be displayed per row and column.

4.5.2. Summary

Extends: JxfsType

Property	Type	Access	Initialized by
JxfsTIOResolution(int columns, int rows)	constructor		-
set*Property(int value)* throws JxfsException	void		sets the corresponding property
get*Property()* throws JxfsException	int		gets the corresponding property
columns	int	RW	device service
rows	int	RW	device service

4.5.3. Properties

columns Property RW

Type *int*
Initial Value -
Description Returns the number of character per column.
Events No additional events generated.
Exceptions No additional exceptions thrown.

rows Property RW

Type *int*
Initial Value -
Description Returns the number of characters per row.
Events No additional events generated.
Exceptions No additional exceptions thrown

4.6. JxfsTIOStatusSelectorEnum Enumeration

This enumeration class is used for the base *getStatus(java.util.List)* method.

Extends: JxfsStatusSelectorEnum

Field	Returned Type	Description
status	JxfsTIOStatus	Status of the text input/output device.

.

4.7. IJxfsTIOConst

4.7.1. Introduction

This interface defines all TIO specific constants. For common constants please refer to the J/XFS Base Architecture.

4.7.2. Constants

Device specific operationID sent with events:

Value	Meaning
JXFS_O_TIO_BEEP	Indicates the *beep* operation completed with an error.
JXFS_O_TIO_LIGHT	Indicates the *lightDisplay* operation completed with an error.
JXFS_O_TIO_LED	Indicates the *setLED* operation completed with an error.
JXFS_O_TIO_DISPLAY	Indicates the *writeDisplayData* operation completed with an error.
JXFS_O_TIO_READ	Indicates the *readKeyboardData* operation completed with an error.
JXFS_O_TIO_CLEAR	Indicates the *clearScreen* operation completed with an error.

Status Event codes:

Value	Meaning
JXFS_S_TIO_STATUS_CHANGED	The status has changed.

Device specific error codes:

Value	Meaning
JXFS_E_TIO_BEEP	Indicates the *beep* operation completed with an error.
JXFS_E_TIO_LIGHT	Indicates the *lightDisplay* operation completed with an error.
JXFS_E_TIO_LED	Indicates the *setLED* operation completed with an error.
JXFS_E_TIO_DISPLAY	Indicates the *writeDisplayData* operation completed with an error.
JXFS_E_TIO_READ	Indicates the *readKeyboardData* operation completed with an error.
JXFS_E_TIO_CLEAR	Indicates the *clearScreen* operation completed with an error.

Method specific constants:

Value	Meaning
JXFS_TIO_BEEP_OFF	The beeper is turned off.
JXFS_TIO_BEEP_KEYPRESS	The beeper will sound on key press.
JXFS_TIO_BEEP_CONTINUOUS	The beeper sounds continuously.
JXFS_TIO_BEEP_EXCLAMATION	The beeper sounds an exclamation signal.
JXFS_TIO_BEEP_WARNING	The beeper sounds an warning signal.
JXFS_TIO_BEEP_ERROR	The beeper sounds an error signal.
JXFS_TIO_BEEP_CRITICAL	The beeper sounds an critical error signal.

Value	Meaning
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.
JXFS_TIO_LED_ERROR	The error LED will be lighted.

JXFS_TIO_LED_WARNING	The warning LED will be lighted.
JXFS_TIO_LED_ONLINE	The online LED will be lighted.
JXFS_TIO_LED_OFFLINE	The offline LED will be lighted (or the online LED turns off).
JXFS_TIO_LED_NORMAL	Indicates proper working of the device
JXFS_TIO_LED_PAPERLOW	The paper low LED will be lighted.
JXFS_TIO_LED_PAPEREMPTY	The paper empty LED will be lighted.
JXFS_TIO_LED_PAPERJAM	The paper jam LED will be lighted.
JXFS_TIO_LED_TONERLOW	The toner low LED will be lighted.
JXFS_TIO_LED_TONEREMPTY	The toner empty LED will be lighted.

Value	Meaning
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

Value	Meaning
JXFS_TIO_TEXT_NORMAL	Normal text.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

The above values are combinable (bitwise OR-able).

Value	Meaning
JXFS_TIO_ECHO_TEXT	The input will be echoed.
JXFS_TIO_ECHO_INVISIBLE	The input will not be echoed.
JXFS_TIO_ECHO_PASSWORD	The input will echo a replacement character.

Keyboard data capabilities

Value	Meaning
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.

The above values are combinable (bitwise OR-able).

Keyboard data output key definitions

Value	Meaning
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT	The (.) sign.
JXFS_TIO_KEY_COMMA	The (,) sign.
JXFS_TIO_KEY_SEMICOLON	The (;) sign.
JXFS_TIO_KEY_FENCE	The (#) sign.
JXFS_TIO_KEY_MULTI	The (*) sign.
JXFS_TIO_KEY_SLASH	The (/) sign.
JXFS_TIO_KEY_PLUS	The (+) sign.
JXFS_TIO_KEY_MINUS	The (-) sign.
JXFS_TIO_KEY_DELETE	The delete key.
JXFS_TIO_KEY_CANCEL	The cancel key.
JXFS_TIO_KEY_ENTER	The enter key.
JXFS_TIO_KEY_CLEAR	The clear key.
JXFS_TIO_KEY_HELP	The help key.
JXFS_TIO_KEY_F1 ... F10	The function keys.

4.8. Constant Definitions

4.8.1. Beep modes

Constant	Numerical Value
JXFS_TIO_BEEP_OFF	8001
JXFS_TIO_BEEP_KEYPRESS	8002
JXFS_TIO_BEEP_CONTINUOUS	8003
JXFS_TIO_BEEP_WARNING	8004
JXFS_TIO_BEEP_ERROR	8005
JXFS_TIO_BEEP_CRITICAL	8006
JXFS_TIO_BEEP_EXCLAMATION	8032

4.8.2. LED modes

Constant	Numerical Value
JXFS_TIO_LED_OFF	8007
JXFS_TIO_LED_CONTINUOUS	8008
JXFS_TIO_LED_SLOWFLASH	8009
JXFS_TIO_LED_MEDIUMFLASH	8010
JXFS_TIO_LED_QUICKFLASH	8011

4.8.3. LED indexes

Constant	Numerical Value
JXFS_TIO_LED_ERROR	- 1
JXFS_TIO_LED_WARNING	- 2
JXFS_TIO_LED_ONLINE	- 3
JXFS_TIO_LED_OFFLINE	- 4
JXFS_TIO_LED_NORMAL	- 5
JXFS_TIO_LED_PAPERLOW	- 6
JXFS_TIO_LED_PAPEREMPTY	- 7
JXFS_TIO_LED_PAPERJAM	- 8
JXFS_TIO_LED_TONERLOW	- 9
JXFS_TIO_LED_TONEREMPTY	- 10

4.8.4. Text positioning modes

Constant	Numerical Value
JXFS_TIO_POS_RELATIVE	8012
JXFS_TIO_POS_ABSOLUTE	8013

4.8.5. Text attributes

Constant	Numerical Value
JXFS_TIO_TEXT_NORMAL	0
JXFS_TIO_TEXT_UNDERLINED	1
JXFS_TIO_TEXT_INVERTED	2
JXFS_TIO_TEXT_FLASH	4

4.8.6. Echo modes

Constant	Numerical Value
JXFS_TIO_ECHO_TEXT	8014
JXFS_TIO_ECHO_INVISIBLE	8015
JXFS_TIO_ECHO_PASSWORD	8016

4.8.7. Key types

Constant	Numerical Value
JXFS_TIO_KEY_NUMERIC	8
JXFS_TIO_KEY_HEXADECIMAL	16
JXFS_TIO_KEY_ALPHANUMERIC	32
JXFS_TIO_KEY_FUNCTION	64
JXFS_TIO_KEY_CONTROL	128
JXFS_TIO_KEY_MISC	256

4.8.8. Error codes

Constant	Numerical Value
JXFS_E_TIO_BEEP	8017
JXFS_E_TIO_LIGHT	8018
JXFS_E_TIO_LED	8019
JXFS_E_TIO_CLEAR	8020
JXFS_E_TIO_DISPLAY	8021
JXFS_E_TIO_READ	8022

4.8.9. Operation Complete codes

Constant	Numerical Value
JXFS_O_TIO_BEEP	8023
JXFS_O_TIO_LIGHT	8024
JXFS_O_TIO_LED	8025
JXFS_O_TIO_CLEAR	8026
JXFS_O_TIO_DISPLAY	8027
JXFS_O_TIO_READ	8028
JXFS_O_TIO_GETKEYS	8030
JXFS_O_TIO_GET_SUPPORTED_KEYS	8031

4.8.10. Status Event codes

Constant	Numerical Value
JXFS_S_TIO_STATUS_CHANGED	8029

4.8.11. Key definitions

Constant	Numerical Value
JXFS_TIO_KEY_0	0
JXFS_TIO_KEY_1	1
JXFS_TIO_KEY_2	2
JXFS_TIO_KEY_3	3
JXFS_TIO_KEY_4	4
JXFS_TIO_KEY_5	5
JXFS_TIO_KEY_6	6
JXFS_TIO_KEY_7	7
JXFS_TIO_KEY_8	8
JXFS_TIO_KEY_9	9
JXFS_TIO_KEY_A	10
JXFS_TIO_KEY_B	11
JXFS_TIO_KEY_C	12
JXFS_TIO_KEY_D	13
JXFS_TIO_KEY_E	14

JXFS_TIO_KEY_F	15
JXFS_TIO_KEY_DOT	17
JXFS_TIO_KEY_COMMA	18
JXFS_TIO_KEY_SEMICOLON	19
JXFS_TIO_KEY_FENCE	20
JXFS_TIO_KEY_MULTI	21
JXFS_TIO_KEY_SLASH	22
JXFS_TIO_KEY_PLUS	23
JXFS_TIO_KEY_MINUS	24
JXFS_TIO_KEY_DELETE	25
JXFS_TIO_KEY_CANCEL	26
JXFS_TIO_KEY_ENTER	27
JXFS_TIO_KEY_F1	28
JXFS_TIO_KEY_F2	29
JXFS_TIO_KEY_F3	30
JXFS_TIO_KEY_F4	31
JXFS_TIO_KEY_F5	32
JXFS_TIO_KEY_F6	33
JXFS_TIO_KEY_F7	34
JXFS_TIO_KEY_F8	35
JXFS_TIO_KEY_F9	36
JXFS_TIO_KEY_F10	37